From Desktop to Mobile:
UI Patterns for User Interface Adaptation in Games

Eveliina Pihlajamäki

University of Tampere
School of Information Sciences
Human-Technology Interaction
M.Sc. thesis
Supervisor: Poika Isokoski
June 2016

University of Tampere, School of Information Sciences
Human-Technology Interaction
PIHLAJAMÄKI, EVELIINA: From Desktop to Mobile: UI Patterns for User Interface Adaptation in Games
M.Sc. thesis, 39 pages, 2 index pages
June 2016

Abstract

Over the last decade, the popularity of mobile gaming has increased dramatically. It is becoming increasingly more common to include mobile as one of the target platforms in game releases. However, implementing a game for multiple platforms and devices is complicated due to the various interaction mechanisms. The versatility with platforms and devices poses major challenges for UI designers; various aspects from resolutions to input devices have to be taken into account. However, by looking at the research in this area it seems that guidelines for multi-platform game UI design have yet to be established, which is what this thesis seeks to address.

Firstly, this thesis investigates whether there are factors in modern gaming applications that differ by device when comparing phone, tablet and desktop user interfaces. For this, three games were picked as case study subjects, and qualitative approach was used to analyze what those factors are and how they are addressed in the UI design of different device versions of the chosen games. The second objective was to form a collection of straightforward design patterns based on the discovered UI differences. As a conclusion, by analyzing several games along the existing research, it is possible to form a collection of design patterns. A set of 17 UI design patterns were documented for adapting a desktop optimized game application to mobile platform.

Key words and terms: user interfaces, ui adaptation, ui design, design patterns, games

Table of Contents

1. Introduction

Over the last decade, mobile gaming has dramatically increased in popularity. As the sales of gaming consoles in the recent years have been decreasing, the number of tablet and smartphone users has been consistently rising together with the sales of PC gaming hardware (Statista, 2016a, 2016b, 2016c, 2016d). Of all applications on mobile platforms, the majority of revenues now comes from games (Takahashi, 2016), which has led to mobile being the most important and largest digital gaming platform by market share for the gaming industry (Super Data, 2016).

The term platform in this thesis means the different hardware environments for smartphones, tablet computers and personal computers. For example, Google Android smartphone is considered a mobile platform, similarly to Apple OSX laptop being considered a desktop platform. Additionally, among the mobile platforms, there may be multiple devices such as smartphones and tablet computers. Multi-platform and multi-device is used interchangeably in this thesis, referring to a piece of software that runs on more than one platform or device.

Due to the aforementioned versatility with devices, it is infeasible for a lot of game studios to focus on releasing games on a single platform such as desktop computer. With cross platform games, it is easier to reach a wider audience and larger revenues. However, implementing a game for multiple platforms and devices is not simple. Using the same user interface on each platform and device is impossible because of the different interaction mechanisms: Mobile interaction is based on touch whereas desktop utilizes mouse and keyboard, and gaming consoles, such as Sony PlayStation or Microsoft Xbox, are controlled by gamepads. All of these interaction mechanisms together with screen size constraints require different ways of providing functionalities and information on the user interface.

In addition to screen differences, there are more aspects to consider outside the desktop environment. First of all, the divergent mobile platforms e.g. iOS, Android and Windows define their own application characteristics. Second, there are various hardware manufacturers using their custom Android versions e.g. HTC, Samsung and One Plus, and third, phone and tablet platforms such as Apple iPhone and iPad require special tailoring from applications. Besides the aforementioned constraints, the entire interaction mechanism will change when converting a desktop game to mobile environment. Mouse and keyboard offer a wide selection of controls, but on a touch device there are considerably fewer options.

Dehlinger and Dixon agree that one of the major challenges in mobile application software engineering concerns creating universal user interfaces. Developers experience

problems particularly with varying screen sizes and resolution. Since all manufacturers have their own standards with screen sizes, the target devices have to be chosen early on in the projects. (Dehlinger and Dixon, 2011)

While designing games with the goal of releasing them on multiple platforms, the UI needs for these devices have to be addressed. In order to maintain a high level of usability and player experience across all the game versions, the UI requires a number of manual alterations per platform.

Naturally, there are design instructions and heuristics that can be applied to user interfaces universally regardless of the application environment and function (Nielsen, 1990; Allen *et al.* 2010). Google (n.d.), Apple (n.d.) and Microsoft (n.d.) have published design guidelines in order to maintain their standards for applications by third party developers. There are even specific design practices and guidelines to mobile games (Lal, 2013; Scolastici and Nolte, 2013). However, these principles are at high level of abstraction, and not applicable when considering existing game application user interfaces that are in need of a platform adaptation.

As multi-platform and multi-device development is being widely adopted, the versatility with platforms and devices poses challenges for UI designers. It seems that guidelines for multi-device game UI design have yet to be established; therefore, this research will seek to begin filling that gap by looking for design patterns. Design pattern can be defined as a solution to a problem that occurs repeatedly in the design process (Borchers, 2001).

The purpose of this thesis is to analyze the user interfaces of chosen games. The aim is twofold; firstly, to identify if there are any differentiating factors in the user interfaces between two device versions of multi-platform games, and to observe and list what the factors are; and secondly, to detect possible design patterns within the game adaptations that were analyzed. The research questions are as follows:

1. *What are the factors in modern gaming applications that differ by device when comparing phone, tablet and desktop user interfaces, and how are they addressed in the UI design of existing multi-platform games?*

The second research question is based on the results of the previous question.

2. *What mobile game UI design patterns can be found for adapting a desktop optimized game application to mobile platform?*

The resulting list of patterns in this thesis is meant to aid game UI designers, and to speed up the development of cross platform games.

This thesis is constructed as follows. This chapter serves as the introduction to the topic of game UI design and presents the research questions. The second chapter will go through the related research and previous work within the field of UI design. The third chapter concentrates on the research methodology and game introductions. The fourth chapter will present the detailed game analysis observations. The fifth chapter is used to form the actual design patterns for multi-device game UI design. The final chapter will sum up and discuss the findings of this thesis. Furthermore, the potential shortcomings of this research and ideas for future research will be discussed.

2. Related Work

Designing user interfaces for different platforms is a complex task. A lot of UI design research has been conducted, dating back almost 30 years (Nielsen, 1990). However, since that time, new technologies and usage contexts have forced designers and developers to take countless of new design factors into consideration. In addition to the very well-known heuristic evaluation guidelines by Jakob Nielsen, there are guidelines for creating user interfaces both for games and regular applications (Bjork and Holopainen, 2004; Fox, 2004; Allen *et al.* 2010; Lal, 2013; Scolastici and Nolte, 2013). This chapter introduces some of the design principles and heuristics, along with different aspects and trends in the area of UI design such as multi-platform design, adaptive and responsive design and UI automation.

The first section will introduce the general topic of what the term game design means and what it entails, and the following sections will discuss the individual approaches to the topic. Lastly, the literature review is summarized.

2.1. What Is Game Design?

Usability is defined in different ways depending on the type of software. For instance, the goal of utility software is to provide a usable, learnable interface with which the user can perform tasks in a specific context of use as efficiently as possible (ISO, 1998). For a game, the purpose is to provide entertainment and challenge, offering a convenient and reliable UI so that the player can enjoy the gameplay without having to concentrate on the user interface (Korhonen and Koivisto, 2006).

For Bjork and Holopainen, game design is like architecture. "The focus of architects is on the intended use of a place and the experiences people should have when crossing a bridge or being in a skyscraper, while the engineers concentrate on details such as load bearing and structural fatigue" (Bjork and Holopainen, 2013, p. 33). Similarly, Brathwaite and Schrieber (2008) divide game design into programming and game design as an art form. In their book Challenges for Game Designers, game design as an art form is further separated into 6 categories.

The first category is *world design*, the creation of story and setting. Secondly, *system design* is the creation of game mechanics, the core game loop and game rules, which is the main component in player engagement. Thirdly, there is *content design*, which is the creation of characters, puzzles and missions around the game mechanics. The fourth type is *game writing*, including dialogue, text and detailed game story. Fifth, there is *level design*, including the progressive level challenges, layouts and maps. Finally, there

is *UI design*, which consists of player interaction with the game, together with feedback and information from the game. (Brathwaite and Schrieber, 2008)

Lal (2013) provides a set of guidelines for the last category, game UI design, in mobile context. The guidelines emphasize "quick entertainment on the go", which means that the flow from one screen to another should be swift, and the game should at all times take into account the changing user environment. The guidelines consist of nine points, divided into four design guidelines and five user experience bullets. The design guidelines include having an option for quick play/pause, having the game in full screen mode with no obtrusive UI controls, using transparent buttons for Pause and Volume, and utilizing device hardware capabilities. User experience guidelines include having the game load quickly, allowing quick access to Pause and Save, having auto save and sleep mode, having auto pause when device notification shows up, and saving preferences information for next game (Lal, 2013). Lal also highlights that the game UI should be fun and rich, which as a general guideline can be considered extremely ambiguous. On the other hand, the first three of the guidelines listed above are truly practical and straightforward, close to what this thesis is aiming at, even though the overall number of the guidelines is fairly low.

2.2. Heuristics and Principles

In addition to complying with detailed design guidelines, UI design can also be approached from a wider angle. Heuristics and principles offer exactly that, a birds-eye view to the general experience of the application. Heuristic usability evaluation is an analysis of the entire user interface conducted by several usability specialists. It is done by following a set of established usability design principles called usability heuristics. It is one of the most well-known methods for evaluating the usability of a software.

Generally, a heuristic model provides a guideline for a high level of usability in utility software. Similarly, playability heuristics do the same for games. However, heuristics are not meant to be absolute, and breaking the guidelines for well-founded reasons is acceptable. Heuristics can be used in both evaluating a piece of software and as a design tool. (Korhonen and Koivisto, 2006)

Korhonen and Koivisto introduced a set of 29 playability heuristics that are specifically designed for evaluating mobile games. The collection is divided into three categories: *Game Usability* covers aspects regarding user interface and game controls. *Mobility* includes issues related to the portable characteristics of the game, and *Gameplay* section concerns the game mechanics and story. (Korhonen and Koivisto, 2006)

In terms of this thesis, the intriguing part is the first module, Game Usability, which refers to a UI that allows fluent game control and displays all the necessary feedback about the game status and actions. The heuristics are:

- Audio-visual representation supports the game
- Screen layout is efficient and visually pleasing
- Device UI and game UI are used for their own purposes
- Indicators are visible
- The player understands the terminology
- Navigation is consistent, logical and minimalistic
- Control keys are consistent and follow standard conventions
- Game controls are convenient and flexible
- The game gives feedback on the player's actions
- The player cannot make irreversible errors
- The player does not have to memorize things unnecessarily
- The game contains help (Korhonen and Koivisto, 2006)

These playability heuristics were created in 2006, and are slightly dated, but they are still logical and applicable to current mobile games. However, as heuristics tend to be very broad in order to be as universal as possible, they are noticeably abstract in a way that they do not, for the most part, answer the question "how to implement this heuristic?" Rather than offering a design solution, they serve as descriptions of the possible challenges to overcome on the UI.

More recently, Sweetser *et al.* (2012) have provided a set of heuristics for the design and evaluation of one specific game genre, real-time strategy (RTS) games. Using the GameFlow model (Sweetser and Wyeth, 2005), they analyzed four RTS game reviews which resulted in 165 heuristics for the elements of Concentration, Challenge, Player Skills, Control, Clear Goals, Feedback, Immersion and Social Interaction. The elements of Player Skills, Control and Feedback include a subcategory of Interface and Controls which refer to the design of user interface. However, as with the work of Korhonen and Koivisto, the heuristics by Sweetser *et al.* are also high level guidelines. They predominantly describe what the player should be able to do with the game, what features should be available in the user interface, and that the controls should be simple and intuitive. As such, the heuristics are insignificant in terms of UI adaptation from desktop to mobile. (Sweetser *et al.* 2012)

There are also guidelines that are not restricted to games; many mobile device manufacturers have published their own design principles as guidelines for developers of all mobile applications (Apple, n.d.; Google, n.d.; Microsoft, n.d.). These guidelines

have been used to form a set of 12 heuristics for touchscreen based mobile devices, addressing the following factors:

- Visibility of system status
- Match between system and the real world
- User control and freedom
- Consistency and standards
- Error prevention
- Minimize the user's memory load
- Customization and shortcuts
- Efficiency of use and performance
- Aesthetic and minimalist design
- Help users recognize, diagnose and recover from errors
- Help and documentation
- Physical interaction and ergonomics (Inostroza and Rusu, 2014)

The heuristics by Inostroza and Rusu (2014) understandably overlap with the ones by Korhonen and Koivisto (2006), indicating the importance of these software usability aspects in mobile devices. However, even with more precise definitions having been given to each of these heuristics, they are more or less open to interpretation and the vision of the designer, which is something this thesis aims to alleviate.

2.3. Multi-Platform Design

As mentioned before, device and platform versatility is one of the greatest inconveniences when designing applications for mobile environment. There are always usage differences from one platform to another, and in terms of development and maintenance, it is typically impractical to design personalized experiences for each platform and screen size. (Pandey, 2013)

With a technology such as HTML5, it is possible, even easy, to develop applications that scale according to the target device. Being a markup language for web, however, it is well suited for browser based applications, yet not an optimal choice in native app development (Puder, *et al.* 2014).

2.3.1. Mobile First

There are methods related to multi-platform design, such as Mobile First. Mobile First means going about the UI development, as the name implies, firstly from mobile perspective, and arranging the content and navigation starting from the small devices. Luke Wroblewski (2011) describes Mobile First as a strategy where the essential part

before everything else is agreeing what content matters the most. The same rationale can then be applied to the other delivery channels of the product, and the prioritization is kept regardless of available screen space.

2.3.2. Responsive Design

Responsive design in web environment means adaptation to device environments and screen sizes (Firdaus, 2013). It provides optimal viewing experience across devices and allows users to read and navigate a page effortlessly without having to spend time on zooming, panning, and scrolling. Responsive designs function in desktop environments as well, which can be seen as the content rearranging itself when the browser window is being dramatically resized.

Pandey (2013) implemented the mobile first approach with responsive design when porting an enterprise transaction banking app to a tablet and smartphone environment from a desktop based web application. After conducting a thorough re-design for the content heavy and feature rich application, he discussed the benefits and challenges faced during the project. On one hand, a business can have a service with single code base to help reduce development and maintenance costs, which works very well for content sites. The tradeoff, however, is being unable to optimize performance and device specific experience, which may not be the desired option when aiming to deliver unique experiences to customers. (Pandey, 2013)

2.4. Automated UI Design Tools

The research for developing UI automation for multi-platform applications is fairly versatile. Predating the current trend of HTML5, several interesting automation tools have been presented over the last ten years.

Starting from the earliest, Ding and Litz (2006) presented a framework for creating multi-platform UIs by annotation and adaptation. Contrary to the currently popular view of mobile first, the framework started from a UI that was originally designed for a device with the largest screen size. To use the tool, first a UI developer graphically inserted annotations to it, then the Adaptation Engine created customized UIs for the smaller target device. In general, the customization strategy considered 1) which components should be transformed, 2) which transformation rule should be applied, and finally 3) how to technically perform the transformation. The paper focuses on explaining the process for the third phase, but the UI components and transformation rules are not described, which would have been more important for this thesis in terms of design patterns.

Multi-platform UI automation has been studied by Lin and Landay (2008) as well. They developed Damask, a prototyping tool which converts web applications for different platforms. Damask was created to allow designers to quickly generate prototype UIs both on PC and mobile phone, and to make it possible to explore and improve their designs at the same time. To use it, a UI design sketch for one device needed to be created, then the tool generated the UI for another device, after which the designer could refine the UIs. In terms of this thesis, the interesting aspect in Damask is that it contained and used a set of design patterns for desktop, web and voice UIs. However, as with the previously mentioned research, the patterns are not disclosed in this paper either.

Jelly (Meskens *et al.* 2010) and Gummy (Meskens *et al.* 2008) are examples of design tools that mix automation with manual design work as well. Both are multi-device UI builders for supporting the design of cross-platform user interfaces. Gummy is a sequential design tool that transforms the original UI design to another platform, after which designers can refine the design further. Gummy was used as a reference when Meskens *et al.* (2010) wanted to learn about the needs for supporting multi-device UI design from professional designers. As the result, they created Jelly, a design environment where UIs can be designed for multiple platforms in parallel. With Jelly it is possible to copy elements from one device design canvas to another and edit the native user interface while preserving the content. While Jelly has similar goals and context for the design adaptation as this thesis, it is not in the scope of this paper to explain in detail the process of converting a UI from one device to another.

2.5. Summary

The previous work in the field of UI design is primarily focused on utility applications rather than addressing specific gaming applications on certain platforms and devices. It does not describe in detail how to adapt a native UI from one device to another, specifically one that involves gaming. Despite presenting high level heuristics and optimized tools for aiding the UI design process, the previous research fails to explain the individual functionalities that appear to be different when using the same native application on various devices, such as a smartphone or a desktop computer. Hence, with the exception of a couple of mobile game UI guidelines by Lal (2013), there is very little contribution available within the academic literature to benefit the goal of this thesis. Presumably, a collection of design changes needed in adapting a game UI would be beneficial for the development, by speeding up the project considerably when no automation in the UI development is used.

3. Method

The methodological approach chosen is of a qualitative nature. Here the process of the qualitative method is, firstly, to choose several games as case study subjects, and secondly, to describe what can be observed as the differentiating factors in the various game versions. Furthermore, the resulting factors are described and interpreted as common design patterns.

3.1. Design

For the case studies, three native game applications were selected as the research subjects. The games are *Minecraft*, *FTL: Faster Than Light* and the original *Angry Birds*.

There were several criteria when selecting the games. First of all, each game should have multiple platform versions: smartphone, tablet and/or desktop computer version. The purpose of having multiple versions was to be able to compare them with each other and to discover any UI features that have been implemented differently in order to adapt the game to the constraints and conventions of the target platform. Initially, a gaming console was considered to be a part of this research, however, it was de-scoped in the very beginning due to lack of resources.

Another criteria was to have a sampling of games that were critically and commercially successful. As the games themselves have succeeded in the market, it is assumed that the design solutions made by the design teams have been proven feasible as well. These solutions are most likely carefully refined, and could be something that can be reapplied and learned from.

The third criteria was to include different genres of games, both serious and casual, strategy and puzzle, in order to not mistake genre specific design decisions as patterns. This criterion would have been better fulfilled had there been more resources to include a larger number of games.

After selecting the games, they were played on the combination of some of the following devices:

- Personal computer MacBook Pro with software version OSX El Capitan 10.11.3
- Tablet computer iPad Mini with software version iOS 9.3.1
- Smartphone OnePlus One with software version Android 5.1.1

Minecraft was played on two devices: on a PC with a downloadable application for Mac computer, and on a smartphone with Minecraft Pocket Edition application downloaded

from the Play Store. FTL: Faster Than Light was played on two devices: on a PC with a downloadable application for Steam platform, and on a tablet with an application downloaded from the App Store. Angry Birds was played on three devices: on a smartphone with an application downloaded from the Play Store, on a tablet with an application downloaded from the App Store, and on a PC with an application downloaded from the App Store.

While playing the game versions side by side with different devices, various factors were observed. The menus, various game views, interaction differences and hardware specific exceptions were listed by game and screenshots were saved. Each observation about the differences were given an identifier, and after listing all observations with identifiers, they were used to locate common patterns. Observations that appeared in several places were combined to a pattern by linking multiple identifiers to it.

Finally, 17 most relevant multi-platform game UI design patterns were formed using the list of observations as a foundation. Some extent of generalization had to be applied in the patterns, however, the initial purpose was to retain them as practical and unambiguous as possible, and that goal was accomplished.

This method of data collection was chosen because it is systematic and feasible for the available resources. The selected approach fits the overall research goals well in terms of searching for common patterns, and ultimately, no alternative approach was appropriate for a systematic UI analysis such as this.

The limitations that can be identified in this research concern the depth of the analysis as well as the quantity of the research subjects. Taking into consideration the vastness of the game Minecraft, in particular, with all of the possible game modes, as well as the exclusion of console versions, it was not possible to analyze all features that the games have to offer. Therefore, only the core features were included from each game.

3.2. Materials

This section introduces the games used in the study, and their characteristics.

3.2.1. Minecraft

Minecraft is an open world construction game by Mojang, originally released in 2011 (Minecraft Wiki, 2016). It is the most popular PC game of all time with more than 22 million copies sold (Makuch, 2016). The game environment consists of three-dimensional rough pixel art, and the purpose of the game is breaking and placing blocks of various types for protection and shelter against the possible non-friendly creatures.

In addition to survival, the game allows building creative structures and artwork on multi-player servers and single player worlds.

There are plenty of Minecraft specific terms used in the game analysis section concerning the gameplay and user interface. The terms are explained below.

- *Gameplay view*, shown in Figure 1, is the basic view where the player looks at the environment without having the inventory, or any crafting view open. Gameplay view includes a HUD hotbar, experience, health and energy indicators.

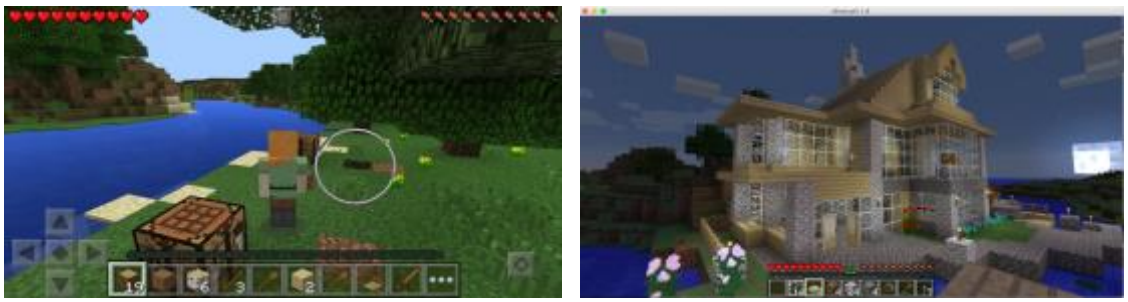

Figure 1: Minecraft's gameplay view on phone and desktop

- *Non-gameplay view* is any other view than the gameplay view, such as inventory or crafting table view.
- *HUD hotbar (Heads Up Display hotbar)* is a line of items which is visible on the gameplay screen at all times. By selecting a slot on the HUD hotbar, the player can quickly switch the item in the avatar's right hand.
- *Item/block* is the core building block in the game world that can consist of various materials. Sometimes a block becomes an item when mined and collected, since an item is not always block shaped. In this section the terms are used interchangeably.
- *Destroying* stands for breaking blocks in order to collect and add them to the inventory.
- *Inventory* is a collection of items the player has mined or found. Any of the inventory items can be placed on the HUD hotbar.
- *Crafting* means producing new items by placing a combination of inventory items on the crafting table. There are two views for crafting in the PC version of Minecraft: the four by four crafting grid on the inventory view, and the three by three crafting grid when using a crafting table item.
- *Using an item* means placing an item block from the HUD hotbar to a specific spot in the game environment. In Minecraft, using an item can also mean eating it or making a block function the way it was intended.

3.2.2. FTL: Faster Than Light

FTL: Faster Than Light, developed by Subset Games, is a single player role-playing game, where the player pilots a spaceship and its crew. As the makers of the game describe it, "the player's goal is to reach a Federation fleet, which is waiting a long distance away, without being destroyed or caught by the rebel fleet in pursuit" (Figure 2). FTL was released on September 14th, 2012. (FTL Wiki, n.d.)

The core gameplay consists of using FTL Drive for jumping to a next location (beacon) on the map, then encountering various random events with different results; the location may be empty, or the player may have to combat an enemy ship, or he may find a store with purchasable upgrades.

Figure 2: FTL beacon map showing player's location on the current sector and rebel fleet in pursuit.

During a fight, for instance, a player can hit Pause and plan his strategic moves without haste. He can select weapons and their targets, assign tasks for the crew and reroute power to different parts of the ship. When he un-pauses the game, the weapons start loading, the shields start operating and the crew members start working (Figure 3). The enemy ship does the same, and after a short while the result of the attack/defense unravels.

Figure 3: FTL gameplay - fight scene on PC and tablet

Figure 4: After jumping to a location in FTL, event dialog is shown with response options.

There are a few FTL specific terms used in the game analysis section concerning the gameplay and user interface. The terms are explained below.

- *Event dialog* is a dialog that appears after each FTL jump, shown in Figure 4. The player is shown an event taking place in the current location and options for reacting to it.
- *FTL Drive* is the mechanism the ship uses to move forward in the game world. The ship uses one unit of FTL fuel for one jump.
- *Gameplay view* is the main view where game events take place with no dialog, map or options view open. Gameplay view includes the ship itself, hull and shields information, indicators for FTL fuel, missiles, drones and scrap currency, FTL charge level, evade and oxygen information, crew health information, crew quick buttons, and systems and subsystems controls.
- *Systems* are the controls on the bottom left of the gameplay view. Systems include reactor, weapons, shields, engines and medbay that heals crew members who are inside the room.
- *Subsystems* are the controls on the bottom right of the gameplay view. Subsystems include piloting, sensors (view of the ship's rooms and enemy ship's rooms) and doors.
- *Powering and depowering* systems means rerouting power from the reactor or any one primary system to another system that is in need of additional power to function efficiently.

3.2.3. Angry Birds

Angry Birds is the most downloaded mobile game ever (App Annie, 2015, Google Play Store, n.d.). It was originally released in 2009 by Rovio Entertainment and has since

evolved into a series of games of the same theme. Angry Birds is a casual game where the player controls and fires birds in a slingshot (Figure 5), in order to destroy egg stealing pigs sitting inside their structures. The goal is to analyze structures and aim to generate chain reactions to take down as many pigs as possible, with a limited number of birds.

Angry Birds differs from Minecraft and FTL in some aspects. Firstly, as opposed to Minecraft and FTL, it was originally designed for a mobile platform, and later ported to PC. Second, the interaction is significantly simpler, consisting only of a single tap and dragging the slingshot. The minimalistic deviances between the different device versions is one of the reasons it was chosen to be a part of this analysis, as to reinforce the core patterns from the other games.

Figure 5: Angry Birds gameplay consists of shooting birds towards pigs with a slingshot.

4. Findings

This section describes the game analysis. For each analysis observation, an identifier has been added consisting of three characters: First letter, second letter and a number (XYN). The first letter represents the game name (M = Minecraft, F = FTL: Faster Than Light, A = Angry Birds). The second letter represents one of the three categories that became evident while analyzing the games: mouse compared to touch (M), keyboard compared to touch (K), and visual adjustments (V). The third character represents the observation order number.

4.1. Minecraft Analysis

This section describes the user interface comparison between the smartphone game version, called Minecraft Pocket Edition, and PC versions of Minecraft. PC Minecraft uses a combination of controls called mouselook/keymove, familiar from most third person games, utilizing mouse and keyboard (Minecraft Controls, n.d.). Because of this, the analysis is divided into three categories:

1. What has been done to replace mouse controls on a phone
2. What has been done to replace keyboard controls on a phone
3. How different views have been adapted to a phone

4.1.1. Mouse vs. Touch

As shown in Figure 1, the main difference between the two game versions is that the touch screen version uses on-screen UI buttons for the gameplay view that the PC version does not have. In the PC gameplay view, the mouse pointer is fixed to the center of the view and cannot be used for clicking the UI buttons. The PC gameplay view indicators are only there for informational purposes. Additionally, the touch screen version provides a multi-touch feature, which makes it possible to utilize simultaneous touch commands and different gestures.

There are five different mouse operations on the PC version of Minecraft; Right click, left click, left long press, scroll and pointer trace. These operations correspond to various actions in the game: using items, clicking on-screen buttons, destroying, selecting items from HUD hotbar and turning/looking. To enable all of the actions on a touch screen, several changes have been made:

- *MM1:* Using item: Right click → Tap
- *MM2:* Attack: Left click → Tap
- *MM3:* Destroy blocks: Left long press → Tap and hold

- *MM4:* Selecting an item from the HUD hotbar: Scroll → Tap on-screen button
- *MM5:* Pointer trace → Finger trace

On the touch screen, a tap gesture is used for both attacking and using an item, as indicated in MM1 and MM2. To enable both actions, they have been made contextual: An attack is performed when a weapon is in hand, and use/place item is performed when a building block is in hand.

4.1.2. Keyboard vs. Touch

On the PC game version there are a lot of configurable operations that can be performed using the keyboard. Naturally this is not possible on a small touch screen without a physical keyboard. Most of the features, however, have been resolved by utilizing various techniques on the touch screen, such as assigning multiple functionalities to an on-screen button e.g. tap and long press.

Keyboard operations correspond to the following gameplay actions: Moving, jumping, opening and closing inventory and chat, sneaking, opening options menu / leaving game, dropping an item to the environment from the HUD hotbar. To enable the actions on a touch screen, several rather significant changes have been made:

- *MK1:* Moving: WASD keys → An additional set of buttons have been introduced to the gameplay view that allow moving to 6 directions (forward, forward right, forward left, right, left, and back)
- *MK2:* Jumping: Space key → Automated as default. Possible to change a setting to use a separate on-screen button.
- *MK3:* Opening and closing a non-gameplay view e.g. Inventory: E key → On-screen button.
- *MK4:* Opening and closing chat: T key → On-screen button.
- *MK5:* Sneaking: Shift key → On-screen button.
- *MK6:* Exiting to main menu: Esc key → Device hard button ‘Back’.
- *MK7:* Dropping an item: Q key → Long press on-screen button.

4.1.3. Menus and Views

A large amount of information and options cannot be incorporated into a small space such as a phone screen. Therefore, the following changes can be observed in Minecraft.

- *MV1:* UI elements on a small screen are relatively large compared to those on a desktop sized display.
- *MV2:* Health, energy and experience indicators are positioned somewhat differently on a small screen, in order to not block the player’s view.

- *MV3:* As shown in Figure 6, menus on a small screen are tabbed while on the desktop version they are hierarchical (main menu) or single page (inventory).
- *MV4:* A game feature (crafting) has been changed dramatically in the touch version compared to the PC version (Figure 7) - Crafting recipes on PC are not shown in game, instead players try and guess them, or alternatively go online and find out new recipes using communities. Instead, on the touch version, the player can see all the items that are currently available for crafting, together with the ones that are not, based on what he has in the inventory (Figure 8).
- *MV5:* Game window size on PC is not fixed, which results in better visibility when a larger window is used.

Figure 6: In Minecraft, Options menu on phone is tabbed and Options menu on desktop is hierarchical, including more configuration options than phone.

Figure 7: In Minecraft, Inventory view on phone does not include quick crafting option, but full crafting is accessible within the same menu.

Figure 8: In Minecraft, avatar, crafting table and inventory are presented on one tabbed menu on phone, whereas PC version has separate views for inventory, avatar and crafting table.

4.2. FTL: Faster Than Light Analysis

This section describes the user interface comparison between the FTL PC and tablet versions. FTL uses mouse and keyboard controls as input. However, it can be played using primarily the mouse.

The main difference between the PC and tablet versions of FTL is the way tooltips and info panels are viewed. Additionally, on PC the available screen space is taken into use by presenting more information in one view, and in contrast, on tablet the available details are reduced to a minimum in order to fit in the necessary UI components.

When adjusting the UI to the tablet version, a few UI components have been repositioned and certain keyboard commands have been replaced with on-screen buttons. Whereas Minecraft utilized multi-touch to adapt a mouse and keyboard control scheme to a touch screen, FTL uses a different approach - various touch gestures, such as long press and swipe.

As with Minecraft analysis, the comparison results are divided into three categories:

1. What has been done to replace the PC mouse controls on a tablet
2. What has been done to replace the PC keyboard controls on a tablet
3. How different views have been adapted from PC to tablet

4.2.1. Mouse vs. Touch

There are four different mouse operations on the PC version of FTL: Left click, right click, hover and click sequence. These operations correspond to various actions in the

game: Viewing system details and tooltips, powering and activating systems, de-powering and de-activating systems, firing weapons and moving crew members.

To enable all of the actions on a touch screen, several changes have been made.

- *FM1:* Event dialog options: On a PC, options in event dialogs can be selected either by left clicking, or with keyboard number keys or the Enter key. → On a tablet, the options have been emphasized by framing them as on-screen buttons, as shown in Figures 4 and 12
- *FM2:* Viewing system details: Mouseover → Swipe right
- FM3: Closing system details: Remove mouseover → Swipe left
- *FM4:* Viewing gameplay tooltips: Mouseover → Long press
- *FM5:* Powering system: Left click → Swipe up
- *FM6:* De-powering system: Right click → Swipe down
- *FM7:* Opening/closing doors: Left click on an individual door or the All Doors -button → Two-phased with on-screen buttons: Activate door system, then open/close door(s)
- *FM8:* Firing weapons: A two phased click sequence; Left click to select weapon, right click to select target → Three phased sequence with on-screen buttons: Open weapons selector, select weapon, select target
- *FM9:* Moving crew members has a similar sequence to that of firing weapons: Two phased click sequence; Left click to select member, then right click to select target room → Two phased sequence with on-screen buttons: select member, then select target room

4.2.2. Keyboard vs. Touch

Due to the keyboard having a secondary control role on the PC version, there are only a few changes that have been made in order to adapt them to the touch screen version. Two additional buttons have been added to the gameplay view (Figure 3), but for an unknown reason the Credits menu has not been altered in that regard.

- *FK1:* Pausing game: Keyboard (Space) → Additional on-screen button (Pause)
- *FK2:* Leaving game: Keyboard (Esc) → Additional on-screen button (Options)
- *FK3:* Closing Credits menu: Keyboard (Backspace) → Long press

4.2.3. Menus and Views

FTL relies heavily on the large number of game controls and there is plenty of information on screen at all times. The tablet version can accommodate the information with some smart alterations to the UI. The following changes has been observed.

- *FV1:* The tablet version has larger buttons than the PC version in relation to screen space
- *FV2:* Gameplay view
    - *FV2a:* Indicator explanation texts (Hull, Shields, FTL Drive, Subsystems) are hidden on a tablet and shown on a PC (Figure 3)
    - *FV2b:* Evade & Oxygen levels indicators are placed differently on a tablet, presumably in order to balance the layout (Figure 3)
    - *FV2c:* Ship systems are ordered differently on a tablet and a PC (Figure 9)
    - *FV2d:* Weapon selector on a tablet is placed vertically not to cover other systems while open, whereas on a PC the selector is placed horizontally in order to not block the view to the ship (Figure 9)
    - *FV2e:* Due to a lack of screen space on a tablet, system selectors are not shown all the time, instead they are opened by tapping the system icon (weapons, doors). The PC version does the opposite, showing system selectors at all times (weapons, doors) (Figures 3 and 9).
    - *FV2f:* Fight mode: Due to a crowded view when two ships are shown side by side on a tablet, it is possible to enlarge either the player's or the enemy's ship, as shown in Figure 3. This feature allows equally accurate control of the fight scene as in the PC version.

Figure 9: FTL weapons selector on PC and tablet. FTL on PC shows weapons selector at all times, whereas player chooses to open and close it on tablet.

- *FV3:* Hangar view (Figure 10)
    - *FV3a:* Additional button in tablet version: Main menu

- *FV3b:* Start game -button placed differently on the tablet version due to the Main menu button taking its original place

- *FV4*: High scores view: The tablet version shows less information on screen as it combines four pages to a single tabbed view, whereas on the PC version there are two blocks of information on screen, and each of them has two tabs (Figure 11)
- *FV5*: Options menu (Figure 12)
  - *FV5a:* The tablet version has considerably less amount of options available
  - *FV5b:* Options have been made to look like buttons on a tablet, whereas on a PC they look like plain text
- *FV6:* Configure controls: This feature applies to PC keyboard controls, therefore, the entire feature is not available on a tablet

Figure 10: Before the game starts on FTL, the player can edit ship properties: The Hangar view on PC and tablet

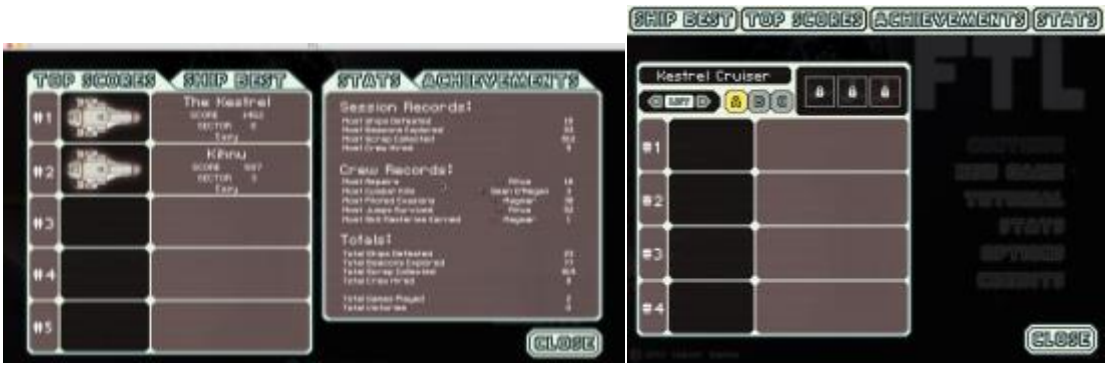

Figure 11: FTL high scores menu on PC and tablet

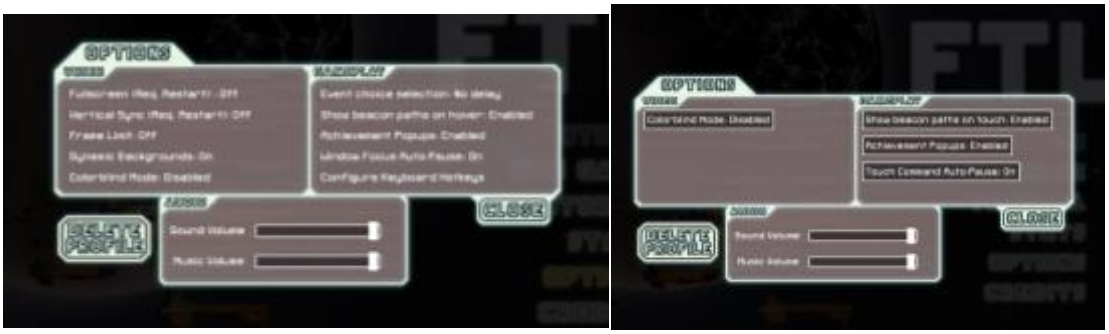

Figure 12: FTL Options menu on PC and tablet

4.3. Angry Birds Analysis

This section describes the user interface comparison between the Angry Birds PC, tablet and phone versions. Angry Birds input on a PC is solely done using the mouse, which results in a low expected number of observations during the analysis.

The comparison results are divided into two categories:

1. What has been done to replace the PC mouse controls on a tablet
2. How different views have been adapted between the devices

4.3.1. Touch vs. Mouse

There are four different touch operations on the mobile versions of Angry Birds: Tap, swipe, 'drag and release' and pinch/pan. These operations correspond to the following actions in the game: Button press, moving the gameplay screen back and forth, slingshot control, and zoom.

To enable touch actions on a PC, the following changes have been made:

- *AM1:* Button press: Tap → Left click
- *AM2*: Move gameplay view right and left: Swipe left and swipe right → swipe left and swipe right (left mouse button)
- *AM3*: Slingshot control: Drag and release → Drag and release (left mouse button)
- *AM4*: Zoom out/in: Pan/pinch → Drag to resize the game window

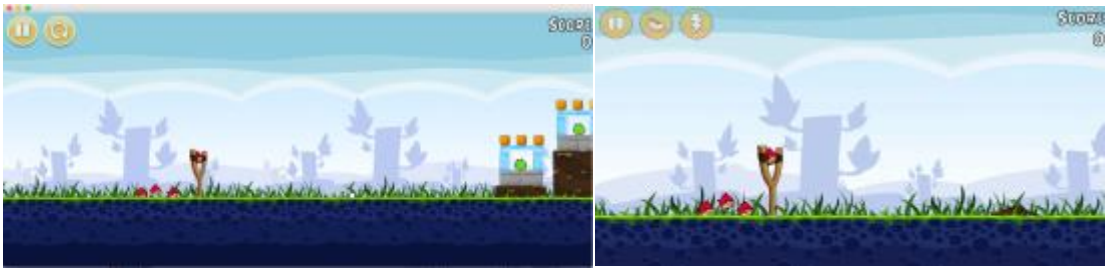

Figure 13: Angry Birds gameplay screen on PC and phone.

In spite of the zoom functionality, control changes were practically non-existent in Angry Birds, which was to be expected with such a straightforward interaction method.

4.3.2. Menus and Views

- *AV1*: The cursor has been customized to look like a hand with a pointing index finger
- *AV2*: Game view can be zoomed out to have better visibility to the target

The only noticeable difference in the menus and views of Angry Birds for phone, tablet and PC concerned varying aspect ratios. On a 4:3 screen, e.g. iPad Mini from the test devices, there was less horizontal space available than on a smartphone's 16:9 screen, leading to less content shown, and narrower default view of the gameplay itself, as shown in Figure 14. Interestingly, this choice of design makes the gameplay more difficult on devices with 4:3 aspect ratio, due to the fact that the player cannot see the target when preparing for the shot, unless choosing to make the game easier by zooming out. Other than that, the user interface was the same across the test devices (Figure 15).

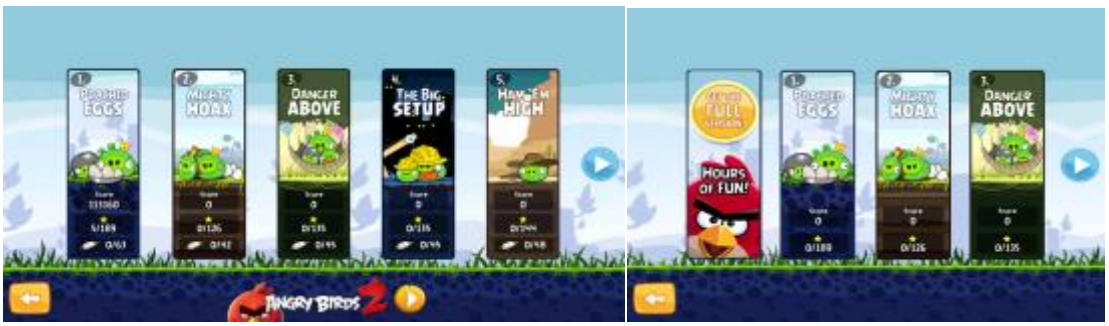

Figure 14: Angry Birds worlds menu on phone and tablet

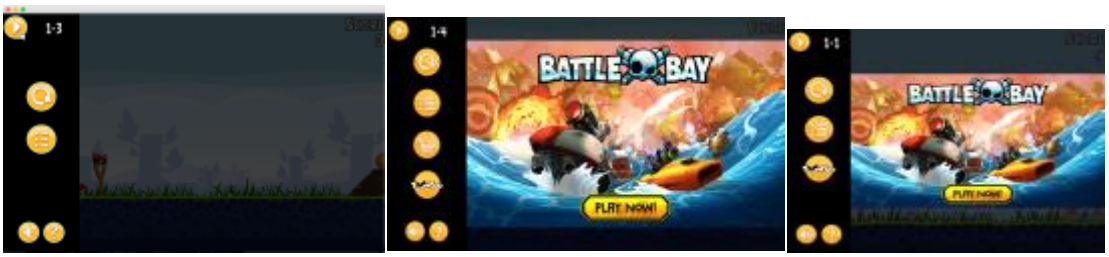

Figure 15: Angry Birds PC, phone and tablet versions do not differ in buttons sizes.

5. Synthesis

Overall, 49 differences were identified, including 17 in Minecraft, 25 in FTL and 6 in Angry Birds. The common differences listed in this chapter form a summary of the game analysis results, and it combines as many observation identifiers into one pattern candidate as possible. The pattern candidate is viewed stronger, the more identifiers it is linked to. The pattern candidates with only one identifier are considered weak, yet due to being an appropriate design choice in the game in question, they are equally regarded as patterns.

The discovered pattern candidates are divided into three categories:

1. Patterns from adapting mouse controls to touch controls
2. Patterns from adapting keyboard controls to touch controls
3. Patterns from adapting views and menus from large to small screens

5.1. From Mouse Controls to Touch Controls

As shown in Table 1, 10 pattern candidates were observed in the selected games regarding the adaptation of mouse controls to a mobile platform and devices with a touch interface. The candidates consist of 20 observations from three different games, the majority of them being considered strong pattern candidates except for the last three.

Pattern candidate	**Observation identifiers**
Left click as the main interaction method across all games is a single tap on touch screens	MM2, FM1, AM1
Mouse long press and mouseover is implemented as tap and hold on touch screen	MM3, AM3, FM4
Left click replaced in touch screen with tap sequence	FM7, FM8, FM9

Right and left click differentiated by context - tapping can have different functions depending on what is happening in the game	MM1, MM2
Mouseover function has been replaced in touch screen with either long press or swipe	FM2, FM3
Left click was considered a positive action (open, power or activate something) and it was replaced with swiping right or swiping up	FM2, FM5
Right click was considered a negative action (close, de-power or de-activate something) and it was replaced with swiping left or swiping down	FM3, FM6
Scrolling to select an item from an on-screen display has been replaced in touch screen with on-screen buttons	MM4
Pointer tracing has been replaced with finger tracing	MM5
Swiping left and right with left mouse key has been replaced with swiping left and right with finger	AM2

Table 1: Pattern candidates from adapting mouse controls to touch controls

5.2. From Keyboard Controls to Touch Controls

As shown in Table 2, six pattern candidates were observed in the selected games regarding the adaptation of keyboard controls to mobile platforms and devices with a touch interface. The candidates consist of 10 different observations from two different games, resulting in three strong pattern candidates and five weak candidates.

Pattern candidate	Observation identifiers
Keyboard commands have been replaced with on-screen buttons	FK1, MK3, MK4, MK5
Using the Esc key for opening menu to exit the game or to edit options has been replaced with pushing the device back button, or with an on-screen button	FK2, MK6
One on-screen button is used for two different keyboard commands, separation is done by short and long press	MK7
Moving to all four directions with keyboard keys (WASD) has been replaced with on-screen arrow buttons that function as a joystick	MK1
Jumping with Space key has been automated in mobile, or alternatively replaced with an on-screen button	MK2
Moving back in menus with Backspace key has been replaced with long press in one occasion	FK3

Table 2: Pattern candidates from adapting keyboard controls to touch controls

5.3. From Large Menus and Views to Small Screen

As shown in Table 3, 10 pattern candidates were observed in the selected games regarding the adaptation of different game views, layouts and menus from large to small screen. The candidates consist of 18 different observations from three games, more than half of them being considered strong and less than half of them weak.

Pattern candidate	Observation identifiers
Button sizes have been increased in small displays in relation to screen size, compared to desktop versions	MV1, FV1
Borders have been added to textual choices to emphasize their functionality as buttons	FV5b, FM1
Indicators and controls have been re-positioned to not block the view of gameplay	MV2, FV2b, FV2c, FV2d
Tabbing has been used in large menus	MV3, FV4
Some features have been removed altogether	FV5a, FV6
Dynamic game window size on PC has been replaced with a full screen fixed size game window on mobile	MV5
Some of the game controls have been hidden on mobile, allowing player to show the controls whenever needed	FV2e
Detailed game components have been made zoomable on mobile	FV2f, AV2
Some textual hints for indicators have been removed	FV2a
Some features have been considerably simplified	MV4

Table 3: Pattern candidates from adapting views and menus from large to small screens

5.4. Patterns

The 26 pattern candidates presented in the previous sections are combined to cover all observations in the analyzed games. In this section, the collection is refined further to

form a condensed set of patterns in adapting a desktop game to mobile platform. Most of the patterns for the condensed list are combinations of strong pattern candidates from the previous sections. Some of the final patterns are unmodified weak pattern candidates, included due to their significance in the game in which they were observed. Moreover, some of the pattern candidates listed in the previous section are excluded from the final list for the opposite reason.

To reinforce the refined pattern collection, four mobile game UI design guidelines from Lal (2013) can be used in a slightly modified form. The guidelines were chosen due to their direct relation to the game user interface and their unambiguous nature.

The first chosen guideline from Lal is having an option for quick Play/Pause. This guideline was actually a feature present in all of the analyzed games, as pressing the Menu button paused the games automatically. Most visibly it can be seen in the game Angry Birds, as shown in Figure 13. In mobile context, the possibility of urgent interruptions is greater than on the more static PC setting, both from the environment and device perspective. Therefore, it is considered a strong pattern to be added to the collection.

The second guideline to be included from Lal was using transparent buttons for Pause and Volume. This guideline, as well, manifests partly in the game Angry Birds in terms of the Pause button, also visible in Figure 13. Regarding having a transparent Volume button on the game UI, Lal contradicts himself with the guideline 'utilize device hardware capabilities', which can be understood as utilizing the device volume hardware buttons for the purpose of adjusting volume, which would be the more logical choice of volume setting in terms of simplicity. Hence, the transparency guideline is added to the pattern collection only partly, excluding the volume statement. On the other hand, the guideline to utilizing device hardware capabilities is merged to the collection as well, as a strong pattern on the grounds that it was observed in the game analysis too.

The fourth guideline to be added to the pattern collection from Lal was having the game in fullscreen mode with no obtrusive UI controls. Since this guideline was found in the game analysis as well, it is used to affirm the first pattern: using the available screen space entirely. The latter part concerning obtrusive UI controls is stated in other words in patterns 5, 13 and 15.

The refined patterns are introduced below. The collection is structured by first presenting the design dilemma in a question format, and then providing the relevant patterns as solutions to the issue.

As with utility applications, should the device status bar be left visible at all times?

1. Occupy the entire screen space for the gaming application, even the status bar is hidden

How to apply primary mouse controls to touch screen?

2. Single tap equals primary mouse click

How to apply keyboard and secondary mouse controls to touch screen?

3. Use on-screen buttons to replace secondary mouse controls and keyboard controls
4. Use on-screen joystick for moving around

How to avoid cluttering the UI?

5. Apply context awareness to tap functionality: the same action performs different functions depending on the gameplay event or context
6. Use a single on-screen button for multiple purposes to reduce the number of necessary buttons: separate functions for short and long press
7. Make static UI components dynamic, player may open and close them as needed
8. Utilize gestures as game controls, i.e. swipe
9. Use swipe gestures as control pairs – assign positive and negative meaning to the opposite directions
10. Use tabbed layout for large menus

How to avoid unnecessary complexity in terms of game controls?

11. Automate less significant parts of the core controls
12. Simplify features or remove some of them altogether

How to utilize mobile hardware UI?

13. Employ device hardware buttons for performing their conventional function: Back, Volume, Mute

How to ensure the usability of the game UI?

14. Enlarge graphical UI components, or let the player zoom into certain areas

How to ensure optimal visibility to the gameplay?

15. Remove unnecessary texts from indicators and buttons, replace with icons
16. Use transparency in UI buttons

How to respect the mobile context?

17. Have an option to quickly pause the game

The goal of this research was to learn if, by examining the literature and existing multi-platform games, a concrete set of design patterns could be found. The patterns presented above are, for the most part, straightforward and understandable. They offer solutions for utilizing mobile platform characteristics to accommodate a large number of UI components that are available in the desktop environment. However, the use of the patterns does not have to be limited to UI adaptation; the patterns are perfectly applicable for creating a brand new game UI design as well.

6. Discussion and evaluation

6.1. Summary

In this thesis, a concept of low level game UI design patterns was introduced. The motivation behind the research goal was to offer tangible ideas for adapting a desktop optimized game to a mobile platform with touch interface. The first research question was:

- *What are the factors in modern gaming applications that differ by device when comparing phone, tablet and desktop user interfaces, and how are they addressed in the UI design of existing multi-platform games?*

In order to approach this challenge, firstly, the previous research in this area was examined. It was found that the topic had not been addressed earlier in this particular context. Instead of practical guidelines for adapting a desktop game user interface to mobile platforms, the existing research concerned creating new application UIs, evaluating UIs for usability and playability, presenting design methods that do not generally offer straightforward guidelines, and tools that can be used to automatically generate user interfaces to target platforms.

An exception to the lack of practical guidelines was made by Lal (2013), mentioned in chapter 2.1. There were four design guidelines in Lal's collection that fulfil the requirement of tangibility and straightforwardness, and complement the pattern set presented in this thesis. In fact, two of the guidelines by Lal were used to complete the pattern set. Furthermore, two of them were discovered during the game analysis and merged in the final list of patterns.

In addition to existing studies, this research question was approached from an angle of a case study where three different games and their desktop and mobile versions were compared and analyzed. The games were Angry Birds, FTL: Faster Than Light and Minecraft. A systematic manual UI comparison was done by analyzing the individual game UI's side by side with desktop computer, tablet and smartphone devices. Various factors were observed: the menus, gameplay views, interaction differences and hardware specific exceptions. All differences between different versions of each game were listed.

Regarding the first research question, it was discovered that a large number of user interface design changes is done when the game is adapted from desktop to mobile platform. There were 48 observations made overall. This is not surprising, considering the complexity of the game UI that the desktop interaction mechanics together with the

large screen size allow. Due to the opposite being true on mobile devices with smaller screens, it was observed that some features were simplified or removed altogether. This was done on the more complex UIs of FTL and Minecraft, both having been adapted with similar design choices.

The second research question concerned the nature of the differences found in the game analysis:

- *What mobile game UI design principles can be found for adapting a desktop optimized game application to mobile platform?*

This question was answered by documenting all of the observations from the game analysis. Initial pattern candidate collection was formed combining as many observations to one pattern as possible. At this point, there were 26 pattern candidates, including all observations regardless of their significance. The observations covered differences in the interaction mechanics, e.g. how to adapt mouse and keyboard controls to touch interface, as well as layout related graphical changes in the game user interface. The final condensed collection includes 17 patterns listed in the previous chapter, all having been minimally generalized in order to not appear too abstract. The resulting patterns proved rational, and excluding the gap in previous research in this area, no surprises were encountered during the process of creating them.

Regarding the analyzed games, Minecraft and FTL: Faster Than Light contributed to the analysis with a large number of observations, whereas the contribution from Angry Birds was scarce. This was expected, however, due to the game having been designed for mobile in the first place. With such a simple UI, there were very few significant changes made to Angry Birds when adapted to a desktop platform with more possibilities for game controls. Thus, the advantages offered by the desktop platform were not utilized in this particular case, and the need for adapting the UI was minimal.

It was noticed that none of the games in this study made use of certain mobile device characteristics, such as accelerator and gyroscope sensors. Even double tap was absent in the identified patterns. Furthermore, multi-touch was not utilized by FTL at all. Angry Birds used multi-touch with the zoom functionality, and Minecraft as the core control with the player being able to move and turn in tandem. Otherwise, there was a versatile selection of controls used as replacement for mouse and keyboard in the selected games.

As the aim of this thesis was to form a concrete set of design patterns to benefit and speed up game development, and to avoid creating high level guidelines for evaluative purposes, it can be said that the research was successful. Granted that this pattern collection is aimed at a narrow sector in game development, a satisfactory level of

tangibility was reached in forming them, as opposed to the characteristics of playability heuristics. Comparing one of the heuristics by Korhonen and Koivisto (2006), for example "Screen layout is efficient and visually pleasing", and one of the final patterns of this thesis, for example "Use tabbed layout for large menus", the contrast between the clarity of the two is noticeable.

Regardless of the heuristics presented in chapter 2.2 being quite insignificant in terms of this thesis, it might be practicable to use them alongside the patterns discovered here. In fact, by examining these patterns, one may form ideas and mockups about the user interface adaptation, whereas playability heuristics can be used to validate those ideas.

6.2. Limitations

One of the limitations that can be identified in this research concerns the depth of the analysis as well as the quantity of the research subjects. Considering the vastness of, for example, the game Minecraft with all of the possible game modes, as well as the exclusion of console versions, it was not possible to analyze all the features that the games have to offer. Therefore, only the core features were included from each game. Had there been resources to include more games, a larger number of relevant observations could have been found, and a more comprehensive set of patterns created.

Furthermore, depending on the selection of games, the set of resulting patterns might look entirely different; even conflicting patterns could be found that were not discovered in these particular games. In fact, this scenario becomes probable if comparing more games in the same genre. Similar games may deviate from the most common UI practices on purpose for individualistic reasons, or for creating a "twist" from which to be remembered. On the other hand, to avoid irritating players by implementing the user interface unconventionally, the designers may want to follow along the trends and the interaction techniques standardized by the mobile industry's most influential players, such as Apple or Google. This, in turn, would decrease the odds of discovering competing UI patterns. Indeed, oftentimes designers make choices based on the target audience and its existing knowledge about similar wide spread products, which is also possible with the games analyzed in this thesis.

Considering the points above, with this research method it is possible to end up with a collection of suboptimal patterns, if such solutions were used in games with decent sales numbers. In that case, the individual design solution would be included as a pattern, if it was subjectively considered feasible by the researcher. On a related note, the individual patterns introduced in this thesis have not been validated in practice. Thus, at

this point it cannot be stated whether each pattern is an applicable or impractical design solution.

In spite of lacking verification, it is clear that the discovered patterns address the various UI components rather well. The patterns contain information for adapting game controls, menus and interaction changes; all aspects of the user interface. Nevertheless, it is not said that by following these patterns, a perfect game UI adaptation can be created. There are always variables in the game UI and mechanics that require careful thought about the best fitting design solutions. Instead of ensuring an infallible user interface adaptation, these patterns give very tangible ideas about what has worked in the existing commercially successful games, and what kind of changes are worth considering. Moreover, for well-grounded reasons, breaking some of these patterns is completely acceptable.

6.3. Future Research

The results of this thesis are complementary to the academic field of game UI design. However, the game development community is expected to benefit to a greater extent as the patterns can be used as the source for UI adaptation ideas. Moreover, the playability heuristics presented earlier can be employed for verifying the designs.

To validate and understand the feasibility of the pattern collection, they should be tested in practice. Using a combination of the patterns in various gaming projects where desktop optimized games are being adapted to mobile platform, would possibly reveal flaws and improvement needs. Furthermore, it would be interesting to form similar patterns resulting from adapting a desktop game to a gaming console platform, or from adapting a console game to a mobile platform. By looking at the literature relating to this subject, there seems to be room for more research concerning this topic.

References

Allen, S., Graupera, V., & Lundrigan, L. (2010). Pro smartphone cross-platform development: iPhone, blackberry, windows mobile and android development and distribution.

App Annie. (2015). "The Most Popular iPhone and iPad Apps of All Time." Retrieved from *http://go.appannie.com/report-most-popular-iphone-ipad-apps-all-time/*

Apple. (n.d.). Designing for iOS. Retrieved from *https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/index.html#//apple_ref/doc/uid/TP40006556-CH66-SW1*

Bjork, S., & Holopainen, J. (2004). Patterns in game design (game development series).

Borchers, J. O. (2000). A Pattern Approach to Interaction Design.

Brathwaite, B., & Schrieber, I. (2008). Challenges for Game Designers. Boston, MA, USA: Course Technology / Cengage Learning. Retrieved from *http://www.ebrary.com.helios.uta.fi*

Dehlinger, J., & Dixon, J. (2011, October). Mobile application software engineering: Challenges and research directions. In *Workshop on Mobile Software Engineering* (Vol. 2, pp. 2-2).

Ding, Y., & Litz, H. (2006, January). Creating multiplatform user interfaces by annotation and adaptation. In *Proceedings of the 11th international conference on Intelligent user interfaces* (pp. 270-272). ACM.

Firdaus, T. (2013). Responsive Web Design by Example. Olton, Birmingham, GBR: Packt Publishing Ltd. Retrieved from *http://www.ebrary.com.helios.uta.fi*

Fox, B. (2004). Game interface design.

FTL Wiki. (n.d.) Retrieved from *http://ftl.wikia.com/wiki/FTL:_Faster_Than_Light_Wiki*

Minecraft Wiki. (2016). Retrieved from *http://minecraft.gamepedia.com/Minecraft_Wiki*

Minecraft Controls. (n.d.). Retrieved from *http://minecraft.gamepedia.com/Controls*

Google. (n.d.). Material design. Retrieved from *https://www.google.com/design/spec/material-design/introduction.html*

Google Play Store. (n.d.) Angry Birds. Retrieved from *https://play.google.com/store/apps/details?id=com.rovio.angrybirds*

International Organization for Standardization (ISO) 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals, Part 11: Guidance on Usability, 1998.

Inostroza, R., & Rusu, C. (2014, April). Mapping usability heuristics and design principles for touchscreen-based mobile devices. In *Proceedings of the 7th Euro American Conference on Telematics and Information Systems* (p. 27). ACM.

Inostroza, R., Rusu, C., Roncagliolo, S., & Rusu, V. (2013, November). Usability heuristics for touchscreen-based mobile devices: update. In *Proceedings of the 2013 Chilean Conference on Human-Computer Interaction* (pp. 24-29). ACM.

Korhonen, H., & Koivisto, E. M. (2006, September). Playability heuristics for mobile games. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services* (pp. 9-16). ACM.

Lal, R. (2013). Digital Design Essentials: 100 Ways to Design Better Desktop, Web, and Mobile Interfaces. Osceola, WI, USA: Rockport Publishers, 2013. ProQuest ebrary. Web. 3 April 2016.

Lin, J., & Landay, J. A. (2008, April). Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1313-1322). ACM.

Makuch, E. (2016). Minecraft PC Reaches New Sales Milestone. Retrieved May 3, 2016 from *http://www.gamespot.com/articles/minecraft-pc-reaches-new-sales-milestone/1100-6433491/*

Meskens, J., Luyten, K., & Coninx, K. (2010, May). Jelly: a multi-device design environment for managing consistency across devices. In *Proceedings of the International Conference on Advanced Visual Interfaces* (pp. 289-296). ACM.

Meskens, J., Vermeulen, J., Luyten, K., & Coninx, K. (2008, May). Gummy for multi-platform user interface designs: shape me, multiply me, fix me, use me. In *Proceedings of the working conference on Advanced visual interfaces* (pp. 233-240). ACM.

Microsoft. (n.d.). Design and UI. Retrieved from *https://developer.microsoft.com/en-us/windows/design*

Nielsen, J., & Molich, R. (1990, March). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 249-256). ACM.

Pandey, S. (2013, September). Responsive design for transaction banking-a responsible approach. In *Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction* (pp. 291-295). ACM.

Puder, A., Tillmann, N., & Moskal, M. (2014, June). Exposing native device APIs to web apps. In *Proceedings of the 1st International Conference on Mobile Software Engineering and Systems* (pp. 18-26). ACM.

Scolastici, C., & Nolte, D. Mobile Game Design Essentials. Olton, Birmingham, GBR: Packt Publishing Ltd, 2013. ProQuest ebrary. Web. 3 April 2016.

Statista. (2016a). Global unit sales of current generation video game consoles from 2008 to 2015 (in million units). Retrieved April 1, 2016 from *http://www.statista.com/statistics/276768/global-unit-sales-of-video-game-consoles/*

Statista. (2016b). Number of smartphone users* worldwide from 2014 to 2019 (in millions). Retrieved April 1, 2016 from *http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/*

Statista. (2016c). Number of tablet users worldwide from 2013 to 2019 (in billions)*. Retrieved April 1, 2016 from *http://www.statista.com/statistics/377977/tablet-users-worldwide-forecast/*

Statista. (2016d). PC gaming hardware industry revenue worldwide from 2012 to 2017 (in billion U.S. dollars). Retrieved April 1, 2016 from *http://www.statista.com/statistics/268367/forecast-of-the-global-pc-gaming-hardware-market*

Super Data. (2016). Mobile games market. Retrieved March 21, 2016 from *https://www.superdataresearch.com/market-data/mobile-games-market/*

Sweetser, P., Johnson, D., Wyeth, P., & Ozdowska, A. (2012, July). GameFlow heuristics for designing and evaluating real-time strategy games. In *Proceedings of the 8th Australasian Conference on Interactive Entertainment: Playing the System* (p. 1). ACM.

Sweetser, P., & Wyeth, P. (2005). GameFlow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)*, *3*(3), 3-3.

Takahashi, D. (2016). Mobile games hit $34.8B in 2015, taking 85% of all app revenues. Retrieved from *http://venturebeat.com/2016/02/10/mobile-games-hit-34-8b-in-2015-taking-85-of-all-app-revenues/*

Wroblewski, L. (2011). Mobile First.